Applying Answer Set Optimization to Preventive Maintenance Scheduling for Rotating Machinery

Anssi Yli-Jyrä[0000−0003−0731−2114] and Tomi Janhunen[0000−0002−2029−7708]

Tampere University, Tampere, Finland
anssi.yli-jyra@tuni.fi,tomi.janhunen@tuni.fi

Abstract. Preventive maintenance (PM) of manufacturing units aims at maintaining the operable condition of the production line while optimizing the maintenance timing and the loss of productivity during maintenance operations. The lesser studied type of preventive maintenance understands a production line as a single machine with multiple components of different maintenance needs. This is relevant when rotating machinery is deployed, e.g., in the paper and steel industries, in the mass production of raw materials consumed by other businesses. A failure in any stage of the production line has the potential of making the entire machine inoperable and enforcing a shutdown and corrective maintenance costs. This work gives an abstract formalization of PM scheduling for multi-component machines as an optimization problem. To provide a lower bound for the complexity of the optimization problem, we prove that the underlying decision problem is NP-complete for varying-size multi-component machines and scheduling timelines. Besides the formalization, the second main contribution of the paper is due to the practical need to solve the problem in industrial applications: the work gives the first encoding of the PM scheduling problem using Answer Set Optimization (ASO). Some preliminary experiments are conducted and reported to set the scene for further algorithm development.[1]

1 Introduction

Preventive maintenance (PM) complements corrective, failure-driven maintenance and plays a key role when it comes to ensuring resource-efficient and timely production as demanded by global manufacturing and resilient industry.

[1] This version of the contribution has been accepted for publication, after peer review (i.e. the paper have been updated to include all changes resulting from peer review as well as any changes of an academic nature requested by the conference organiser) but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1007/978-3-031-21541-4_1. Use of this Accepted Version is subject to the publisher's Accepted Manuscript terms of use https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms.

While PM brings obvious benefits, the scheduling part of PM is tricky and worth digitalization and optimization. Digitized *PM scheduling* (PMS) is a challenging computational problem. Scheduling for one- or two-component machines has been studied a lot, but research has increasingly shifted to the PMS of multi-component machines with serial or serial-parallel dependencies [9].

Multi-component machines with rotating components are commonly deployed and maintained, e.g., in the dairy, paper and steel industries, for the mass production of raw materials like dairy, newsprint, and rolled steel. Their rotating components, *rotors*, operate in synchrony to form a serial, continuous *production line*, the best example of which is perhaps a paper machine (see Fig. 1).

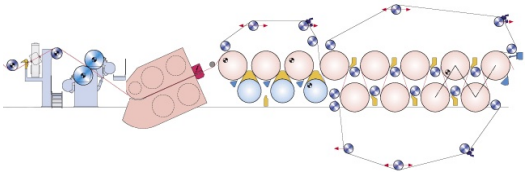


Fig. 1: A paper machine (knowpap.com)

Both preventive and run-to-failure, corrective maintenance policies and respective scheduling are needed for these machines. According to the component-wise failure distribution of paper machines [22], the aging of the components in the first sections of the production line increases their risk of predictable failure. These kinds of components demand time- or condition-based preventive maintenance, while the latter sections of the paper machine demand a corrective maintenance policy and are out of the scope of the current work. Olde Keizer et al. [21] review condition-based maintenance policies for multi-component systems with various dependencies. A particular structural property of serially connected rotating multi-component machines is that their components are *physically dependent* on one another. Not only the independent maintenance of dependent components is restricted, but maintenance of any stage of the machinery often makes the entire production line inoperable [19, 20]. For this reason, we concentrate on the maintenance of components that are relevant in this respect.

Selective maintenance is a PM policy whose purpose is to help select a group of components for maintenance operations so as to optimize the PM schedule with respect to chosen optimization criteria such as *system availability, maintenance cost, life cycle cost*, and *reliability*; as analyzed, e.g., in [6, 18, 23]. Unlike [17], we neither integrate production and maintenance scheduling explicitly nor study concrete machines. Instead, our goal is to investigate options when it comes to the formalization of new kinds of simple optimization criteria. We express a limit for maintenance cost as a limit for maintenance breaks, and we establish new optimization criteria that are somewhat dissimilar to the former criteria. The first criterion, *under-coverage*, measures how often a preventive maintenance action is delayed in the timeline, reducing the reliability of the system. The second criterion, *over-coverage*, measures how often a preventive maintenance action occurs earlier than necessary, increasing the life cycle cost.

According to the survey [17], most approaches to PMS allow no exceptions beyond designated time intervals and flexibility windows around them, but we do not insist on maintenance that is based on such *periodic* recommended main-

tenance intervals of components. Our target for maintaining a particular component is extremely flexible and opportunistic: We do not only allow delays or advances in timing [1, 27], but we allow *arbitrary deviations* from the recommended maintenance interval if this is beneficial to the optimization criteria. Any maintenance break is an opportunity to maintain components ahead [30] of their due time. Such over-maintenance (over-coverage) can reduce the need for separate scheduled maintenance breaks and help to improve system availability.

Hoai and Luong [18] determine a policy for scheduling one maintenance period. This gives a convex, efficient model for availability and cost. You and Meng [26] present a PM scheduling for small multi-component serial machines and a small timeline. Similarly, our goal is to schedule multiple maintenance breaks and operating periods. This unlocks numerous opportunities to optimize the schedule as a whole. As any future state of the machine has a finite model description, periods longer than individual intervals can ultimately emerge in the optimal schedules. However, due to degradation and random changes in the machine condition, the distant future in the schedule is highly speculative. For this reason, our model is intended to be used inside a modularized predictive maintenance framework with real-time sensory-updated prognostic information of the current machine state and a rescheduling loop [26].

Previous implementations of PMS have deployed a myriad of AI techniques, including *genetic algorithms* (GAs) [8], *mixed-integer programming* (MIP) [7], *dynamic programming* [29], and formulations as *constraint satisfaction problems* (CSPs) [12]. In 1975, Zurn and Quintana [29] classified the PMS search techniques into those that search for (i) a local optimum, (ii) a piece-wise approximation, and (iii) a globally optimal, exact solution. Frost and Dechter [12] showed that a CSP solver can be extended to an iterative framework that solves the PMS as an optimization problem. Since then, the performance of exact solvers, including their extensions for solving optimization problems, has improved drastically. It is already known that *answer set programming* (ASP) is well-suited for solving scheduling problems (see, e.g., [3, 10, 11]). The ASP paradigm (see [4] for an overview) offers a rule-based language for *encoding* search problems as logic programs such that solutions are captured by *answer sets*. When an objective function is incorporated in the program, the search process turns into answer set optimization (ASO). However, to the best of our knowledge, there is no prior PMS implementation nor up-to-date feasibility studies based on ASO. The current work fills this gap by presenting two PMS models implemented in ASP and their global, iterative optimization supported by modern ASP solvers.

This work claims three major contributions to the area: (1) We present a highly abstracted PMS formalization. While this ignores many previously studied aspects, such as failure models, resource limits, duration of actions, production targets, it is a part of exploratory, non-incremental research and extensible through additional constraints and a modularized framework. (2) By studying the complexity of the formalization, we connect it to other formalizations with a similar complexity (e.g. [12]). (3) By making the ASP encodings publicly available, we facilitate the systematic improvement of encodings for the problem.

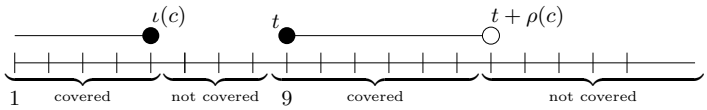


Fig. 2. Illustration of $\iota(c) = 5$ and $\rho(c) = 6$ with a service at time step $t = 9$

The rest of this article is organized as follows. In Section 2, we provide the formal definition of a multi-component machine and its maintenance schedule, and investigate some objective functions that are relevant for the optimization of schedules. These definitions are crucial building blocks in Section 3 where we formalize PMS for multi-component machines from a number of perspectives. The computational complexity of the resulting decision and function problems is then roughly characterized in the same section. To move from the theoretical analysis to practice, we present an ASP encoding of the PMS optimization problem in Section 4. The performance obtained by this encoding for PMS is preliminarily studied in Section 5 using the Clasp solver to implement the actual optimization of schedules. Section 6 concludes the paper.

2 Definitions of Machines and Schedules

In this section, we provide an abstraction of a multi-component machine and a schedule specifying preventive maintenance actions for its components in the scheduling timeline. These definitions pave the way for the definition of the scheduling problem whose variants will be studied further in the next section.

Definition 1. *A* multi-component machine *is a triple* $\mathcal{M} = \langle C, \iota, \rho \rangle$ *consisting of a finite set of* components C*, an* initial lifetime function $\iota : C \to \mathbb{N}$*, and a* recommended maintenance interval *function* $\rho : C \to \mathbb{N} \backslash \{0\}$.

Fig. 2 illustrates how the values of the functions in Definition 1 are to be interpreted. The *initial lifetime* function ι tells how many time steps are covered by maintenance actions performed prior to the timeline for intended PM scheduling. Quite similarly, the *recommended maintenance interval* (RMI) function ρ indicates how many time steps are covered by each preventive maintenance action. Thus, a preventive maintenance action taking place at a time step t starts an RMI that is over at the time step $t + \rho(c)$. Every time t when a component is maintained it becomes as good as new while all non-maintained components continue aging during the time step t. When multiple components and several preventive maintenance actions are taken into consideration and restricted to a finite timeline, the resulting notion of a schedule is detailed as follows.

Definition 2. *A* preventive maintenance schedule *(PMS, or* schedule *for short) for machine* $\mathcal{M}$ *is represented as a quadruple* $S = \langle h, \ell, b, A \rangle$ *where*

- $h \in \mathbb{N} \backslash \{0\}$ *marks the* horizon *of the scheduling timeline,*

- $\ell \in \{1, \ldots, h\}$ *is the* limit *for the last possible time step of a preventive maintenance action,*
- $b \in \{0, \ldots, \ell\}$ *is the* maximum size of a set $B \subseteq \{1, \ldots, \ell\}$ of time steps*, also called scheduled maintenance breaks, during which any preventive maintenance action must take place,*
- $A : C \times \{1, \ldots, \ell\} \to \{0, 1\}$*, called the* service selection function*, is a characteristic function indicating, for each component $c \in C$ of the machine $\mathcal{M}$, the set $B_c = \{t \mid A(c, t) = 1\} \subseteq B$ of time steps of preventive maintenance actions used to service component c.*

The limit ℓ controls how far towards the horizon the breaks can be allocated, having a compressing effect on the schedule. If $\ell = h$, any preventive maintenance action performed at the time step h is mainly a wasted investment in the future without a significant effect on the scheduling timeline. The schedule is *empty* if $b = 0$. Four evaluation functions for the quality of component-wise schedules are defined. Their definitions are facilitated by two auxiliary functions: let $\pi(c) = \{(s, t) \in B_c \times B_c \mid s < t$, and $s < u < t$ for no $u \in B_c\}$ be the pairs of consecutive maintenance times of component $c \in C$ and $\delta : \mathbb{Z} \to \mathbb{N}$ a filter function defined in such a way that $\delta(x) = 0$ when $x < 0$ and $\delta(x) = x$ otherwise.

Definition 3. *For each schedule S, we define the component-wise* over-coverage function $oc : C \to \mathbb{N}$*, the component-wise* under-coverage function $uc : C \to \mathbb{N}$*, the component-wise* miscoverage function $mc : C \to \mathbb{N}$*, and the component-wise* action count function $ac : C \to \mathbb{N}$ *as follows.*

If B_c is empty, we have $oc(c) = ac(c) = 0$, $uc(c) = mc(c) = h - \iota(c)$. Otherwise, the values of the componentwise functions are given by

$$oc(c) = \delta(\iota(c) - \min B_c + 1) + \sum_{(s,t)\in\pi(c)} \delta(\min(s + \rho(c), h + 1) - t),$$

$$uc(c) = \delta(\min B_c - \iota(c) - 1) + \sum_{(s,t)\in\pi(c)\cup\{(\max B_c, h+1)\}} \delta(t - (s + \rho(c))),$$

$$mc(c) = oc(c) + uc(c), \qquad ac(c) = |B_c|.$$

Intuitively, the value $oc(c)$ indicates the number of time steps during which the implementation of a due preventive maintenance action of component c is advanced earlier from the time suggested by the recommended maintenance interval $\rho(c)$. The value $uc(c)$ indicates, for component c, the number of time steps that are neither covered by the initial lifetime nor a recommended maintenance interval started by a preventive maintenance action. The miscoverage $mc(c)$ simply combines these two quality evaluation functions into a sum, and the action count $ac(c)$ tells the number of preventive maintenance actions of component c.

Lemma 1 links the under-coverage, the over-coverage and the number of preventive maintenance actions to each other. Lemma 2 demonstrates that over-coverage is potentially much larger than under-coverage. Finally, Lemma 3 shows that servicing too often does not help to reduce the under-coverage.

Lemma 1. *Let $c \in C$ and assume that $\ell + \rho(c) \leq h$. Then $uc(c) = h - \iota(c) - ac(c)\rho(c) + oc(c)$.*

Proof. Define first the sequence $B_c^0, B_c^1, \ldots, B_c^{ac(c)}$, in such a way that $B_c^0 = \emptyset$, $B_c^1 = \min B_c$, and $B_c^k = B_c^{k-1} \cup \{t \mid (\max B_c^{k-1}, t) \in \pi(c)\}$. This gives us a growing sequence of service times $t_1 = \max B_c^1, \ldots, t_{ac(c)} = \max B_c^{ac(c)}$.

The lemma is now proven by induction on k, $0 \leq k \leq ac(c)$.

$$
\begin{array}{lll}
k = 0\text{:} & & oc_0(c) = 0, \\
 & & uc_0(c) = h - \iota(c) - k\rho(c) + oc_0(c). \\
k = 1\text{: } (i) & \iota(c) < t_1 : & oc_1(c) = 0, \\
 & & uc_1(c) = h - \iota(c) - k\rho(c) + oc_1(c). \\
 (ii) & \iota(c) \geq t_1 : & oc_1(c) = t_1 - \iota(c) + 1, \\
 & & uc_1(c) = h - \iota(c) - k\rho(c) + oc_1(c). \\
k > 1\text{: } (i) & t_{k-1} + \rho(c) \leq t_k : & oc_k(c) = oc_{k-1}(c), \\
 & & uc_k(c) = h - \iota(c) - k\rho(c) + oc_k(c). \\
 (ii) & t_{k-1} + \rho(c) > t_k : & oc_k(c) = oc_{k-1}(c) + (t_{k-1} + \rho(c) - t_k), \\
 & & uc_k(c) = h - \iota(c) - k\rho(c) + oc_k(c).
\end{array}
$$

Thus $uc_k(c) = h - \iota(c) - k\rho(c) + oc_k(c)$ for all $k, 1 \leq k \leq ac(c)$. □

Lemma 2. *Let $c \in C$, and assume that $\ell + \rho(c) - 1 \leq h$ and $\iota(c) = 0$. Then $0 \leq uc(c) \leq h$, and $0 \leq oc(c) \leq (\ell - 1)(\rho(c) - 1)$.*

Proof. Clearly, $uc(c) \geq 0$. On one hand, the value $uc(c)$ reaches 0 when the services of the component c occur at time steps ℓ and $t = \iota(c) + 1 + k\rho(c)$, such that $k \geq 0$ and $t < \ell$. In this way, $uc(c) = h - (\ell + \rho(c) - 1)$. If $\ell = h - \rho(c) + 1$, we reach $uc(c) = h - (h - \rho(c) + 1 + \rho(c) - 1) = 0$. If $\ell > h - \rho(c) + 1$ and $B_c = \{1, \ldots, h\}$, we still have $uc(c) = 0$. On the other hand, the value $uc(c)$ is the greatest when $B_c = \emptyset$. In this case, $uc(c) = h$, and $oc(c) = 0$.

Clearly, $oc(c) \geq 0$. In fact, $oc(c) = 0$ when $B_c = \emptyset$, but the value $oc(c)$ is the greatest when $B_c = \{1, \ldots, \ell\}$. In this case, $oc(c) = (\ell - 1)(\rho(c) - 1)$. □

Lemma 3. *Let w and t such that $w + 2 \leq t \leq w + \rho(c)$ be two maintenance times of component c, contributing $w + \rho(c) - t$ to $oc(c)$. Adding a maintenance time v, such that $w + 1 \leq v \leq t - 1$, increases $oc(c)$ without decreasing $uc(c)$.* □

Proof. Fig. 3 shows a situation described in the lemma. The service at time step v has only an increasing effect on the over-coverage of the schedule. □

The component-wise evaluation functions are lifted for a machine as follows:

Definition 4. *For every schedule $S = (h, \ell, b, A)$, there are associated measures: the* over-coverage $oc(C) = \sum_{c \in C} oc(c)$*, the* under-coverage $uc(C) = \sum_{c \in C} uc(c)$*, the* miscoverage $mc(C) = uc(C) + oc(C)$*, and* action count $ac(C) = \sum_{c \in C} ac(c)$.

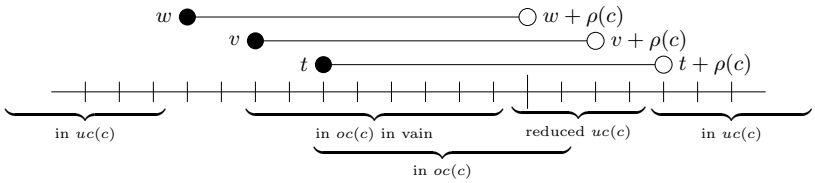


Fig. 3. Servicing too often does not pay off

In the rest of the paper, when solving PMS problems, we employ no other measures of quality than under-coverage and miscoverage. The following two lemmas identify corner cases for these measures.

Lemma 4. *Let $\mathcal{M} = \langle C, \iota, \rho \rangle$ be a multi-component machine. For any horizon $h \in \mathbb{N} \setminus \{0\}$, and limit $\ell \in \{1, \ldots, h\}$, there is a schedule $S = \langle h, \ell, b, A \rangle$ such that the over-coverage $oc(C)$ associated with the schedule is 0.*

Proof. The over-coverage $oc(C)$ of the empty schedule is 0. □

Lemma 5. *Let $\mathcal{M} = \langle C, \iota, \rho \rangle$ be a multi-component machine. For any horizon $h \in \mathbb{N} \setminus \{0\}$, and breakset size $b \geq \lceil h / \min_c \rho(c) \rceil$ there is a schedule $S = \langle h, \ell, b, A \rangle$ such that the under-coverage $uc(C)$ associated with S is 0.*

Proof. Assume without loss of generality that $\iota(C) = \{0\}$. Construct a schedule $\langle h, \ell, b, A \rangle$ such that $B_c = \{1 + i \times (\min_c \rho(c)) \mid i = 0, \ldots, b-1\}$ for all $c \in C$. These preventive maintenance actions are enough to cover the scheduling timeline, giving under-coverage $uc(C) = 0$. □

3 Basic PMS Problems and Their Complexities

In the following, we define some variants of the PMS problem and study their computational complexities.

Definition 5. *The EXACT MISCOVERAGE PMS problem is a decision problem that assumes, as its input, a multi-component machine $\mathcal{M} = \langle C, \iota, \rho \rangle$ and a quadruple $T = \langle h, \ell, b, m \rangle$ of scheduling parameters, and poses the question of whether the machine has a schedule $S = \langle h, \ell, b, A \rangle$ such that $mc(C) = m$.*

The NP membership of EXACT MISCOVERAGE PMS will be shown under the assumption that all the elements of the scheduling timeline $\{1, \ldots, h\}$ are separate units of the input, i.e., h is essentially encoded in unary. The following lemma gives an upper bound for the computation of the miscoverage.

Lemma 6. *Let $\mathcal{M} = \langle C, \iota, \rho \rangle$ be an arbitrary multi-component machine and $S = \langle h, \ell, b, A \rangle$ be one of its schedules. The miscoverage $mc(C)$ associated with S can be computed in $O(|C|h)$ time and $O(|C|h)$ space.*

Proof. Assume accessing of A takes $O(1)$ time steps. We compute the miscoverage of the schedule S with a loop that runs over C, and with an inner loop over all time steps $\{1, \ldots, h\}$ in $O(|C|h)$ time: For each component c, the inner loop starts from time step 1 and keeps track of the initial lifetime and the RMI that starts at a preventive maintenance action. (i) Every extra RMI covering the time step contributes one over-coverage to $mc(C)$. (ii) Each time step not covered by any RMI contribute one under-coverage to $mc(C)$. In addition, it is safe to say that the space requirement of the computation is $O(|C|h)$. □

Theorem 1. *The EXACT MISCOVERAGE PMS problem is in NP.*

Proof. Let an EXACT MISCOVERAGE PMS problem instance consist of a multi-component machine $\mathcal{M} = \langle C, \iota, \rho \rangle$ and parameters $T = \langle h, \ell, b, m \rangle$. Since $\iota(c) < \rho(c)$ for all $c \in C$, and $h \geq \ell \geq b$, the input length is $\Omega(|C| \max_c \rho(c) + h + \log m)$. A certificate to the EXACT MISCOVERAGE PMS problem consists of a schedule $S = \langle h, \ell, b, A \rangle$ that requires $O(|C|h)$ space, which is polynomial in the input size. Let S be an arbitrary certificate. To verify it, we only need to check that the miscoverage $mc(C)$ associated with S equals m. By Lemma 6, $mc(C)$ can be computed in a polynomial time and space. □

The SUBSET SUM problem (SSP) is an example of an NP-complete problem [13]. In the sequel, it will be shown to be at most as hard as the EXACT MISCOVERAGE PMS problem by using an appropriate reduction.

Definition 6. *The SSP is a decision problem that assumes as its instance a pair $\langle N, s \rangle$ where $N = \{c_1, c_2, \ldots, c_n\}$ is a multiset of positive integers $c_i \in \mathbb{N} \setminus \{0\}$, $i = 1, \ldots, n$, and $s \in \mathbb{N} \setminus \{0\}$ is the target sum for a subset of these integers. The problem is to decide whether there is a subset $N' \subseteq N$ such that the target s is obtainable as the sum of the elements of N', i.e., $\sum N' = s$.*

Theorem 2. *The EXACT MISCOVERAGE PMS problem is NP-hard.*

Proof. Let $P = \langle N, s \rangle$, $N = \{c_1, c_2, \ldots, c_n\}$ be an arbitrary instance of SSP. Without loss of generality, assume that $c_i \leq c_{i+1}$, for $1 \leq i \leq n-1$. This SSP instance reduces by a poly-time function to an instance of EXACT MISCOVERAGE PMS given by machine $\mathcal{M} = \langle C, \iota_0, \rho \rangle$ and scheduling parameters $T = \langle \max N, 1, 1, n \max N - s \rangle$, such that $C = \{1, \ldots, n\}$, and $\iota_0(i) = 0$, $\rho(i) = c_i$ for all $i \in C$. Since all preventive maintenance actions are enforced to occur at time step 1, we have $oc(C) = 0$ for all schedules S. If S is the empty schedule, then $uc(C)$ reaches its maximum value $n \max N$.

Let $N' \subseteq N$ be a multiset subset with sum s, being a solution to the SSP instance $\langle N, s \rangle$. In the reduction, each integer $c_i \in N'$ is encoded by $A(i, 1) = 1$, while $A(i, 1) = 0$ encodes that $c_i \notin N'$. This gives a schedule $\langle \max N, 1, 1, A \rangle$ whose associated under-coverage $uc(C)$ is $n \max N - s$. This schedule is a solution to the EXACT MISCOVERAGE PMS instance $\langle \mathcal{M}, T \rangle$.

Conversely, let $S = \langle \max N, 1, 1, A \rangle$ be a solution to the EXACT MISCOVERAGE PMS instance $\langle \mathcal{M}, T \rangle$. The associated under-coverage is $n \max N - s$. The schedule encodes the multiset subset $N' = \{c_i \mid i \in C, A(i, 1) = 1\} \subseteq N$ with sum s. This subset is a solution to the SSP instance $\langle N, s \rangle$. □

Listing 1. A PMS problem instance

```
1 comp(1,5,2).    comp(3,7,0).    comp(5,9,0).    comp(7,5,4).
2 comp(2,10,0).   comp(4,4,3).    comp(6,11,2).   comp(8,8,0).
```

Definition 7. *The BOUNDED MISCOVERAGE PMS problem is a decision problem that assumes, as its input, a multi-component machine* $\mathcal{M} = \langle C, \iota, \rho \rangle$ *and a quadruple* $T = \langle h, \ell, b, m \rangle$ *of bounds, and poses the question of whether the machine has a schedule* $S = \langle h, \ell, b, A \rangle$ *such that* $mc(C) \leq m$.

Theorem 3. *The BOUNDED MISCOVERAGE PMS problem is in NP.*

The proof of Theorem 3 is similar to Theorem 1 and thus left to the reader.

Function problems can be defined almost in a similar way as decision problems. The following function problems concern the optimization of schedules with respect to particular measures associated with them.

Definition 8. *The UNDER-COVERAGE PMS and MISCOVERAGE PMS are optimizing function problems whose inputs consists of a multi-component machine* $\mathcal{M} = \langle C, \iota, \rho \rangle$ *and a triple* $\langle h, \ell, b \rangle$ *of scheduling parameters. The solution to the UNDER-COVERAGE PMS problem is a schedule* $S = \langle h, \ell, b, A \rangle$ *such that the under-coverage* $uc(C)$ *associated with the schedule* S *is minimized, and the solution to the MISCOVERAGE PMS problem is a schedule* $S = \langle h, \ell, b, A \rangle$ *such that the miscoverage* $mc(C)$ *associated with the schedule* S *is minimized.*

4 An ASO-Based Implementation

In what follows, we present an ASP encoding of the PMS problem. The encoding is presented in the language fragment of the Gringo grounder as described by Gebser et al. in [16]. Thus existing ASP solvers such as Clasp [14] and WASP [2] can be readily used to implement the search for optimal schedules in practice.

Listing 1 illustrates the representation of an eight-component machine using the domain predicate `comp(C,R,L)` whose arguments give the identity `C`, the recommended maintenance interval `R`, and the initial lifetime `L` due to a preventive maintenance action before the scheduling timeline. The machine instantiates the PMS problem whose encoding is split into three sections given in Listings 2–4.

In Listing 2 (Line 2), the parameters `h`, `l`, and `b` for the number of time steps, the limit for the last maintenance break, and the number of scheduled maintenance breaks, respectively, are set to their default values. In Line 3, we define `time/1` as a domain predicate for representing time steps. Moreover, the identities of components are extracted as the extension of the `comp/1` predicate in Line 4, recall `comp/3` from Listing 1. Then we are ready to choose time steps for scheduled maintenance breaks, as formalized by the choice rule in Line 7. At most `b` scheduled maintenance breaks are picked first and for each of these breaks at least one component is selected for an preventive maintenance action

Listing 2. PMS encoding: parameters, domains, choices, and service actions

```
1  % Parameters and domains
2  #const h=32.  #const l=32.  #const b=3.
3  time(0..h).
4  comp(C) :- comp(C,_,_).
5
6  % Breaks and the allocation of components for maintenance
7  { break(T): time(T), T>0, T <= l } <= b.
8  1 <= { serv(C,T): comp(C) } :- break(T).
9
10 % End of maintenance interval
11 emi(C,T+R) :- comp(C,R,L), serv(C,T), time(T+R).
12 emi(C,L+1) :- comp(C,R,L), L>0, time(L+1).
```

in Line 8. This is represented in terms of the `serv/2` predicate that eventually captures solutions to PMS problems. In Line 11, we define when the respective RMI ends using the `emi/2` predicate for each such action. For components `C` with some initial lifetime, the analogous time step is defined in Line 12.

Based on the predicates introduced so far, we may define how time steps are covered by RMIs on a component-by-component basis, see Listing 3. This leads to an encoding ('**2-level**') with recursive definitions of two predicates `cov1(C,T)` and `cov2(C,T)` denoting that a component `C` is covered by exactly one or two RMIs, respectively, at time `T`. Naturally, the target is that `cov1(C,T)` would hold for every `C` and `T`, indicating that `C` is neither under- nor over-serviced over time. The base cases for `cov1` are given in Lines 2–4: either `C` has some initial lifetime `L` due to recent maintenance, or `C` is maintained at time `T` not covered by any earlier maintenance actions at the preceding time step `T-1`. Based on this, we may infer that `cov1(C,T+1)` holds for the following time step `T+1`: either `C` is not maintained again nor the RMI ends (Line 6), or `C` is maintained again but the preceding RMI ends at the same time (Line 7). The third possibility is that `C` is maintained again within the same RMI, a reason for making `cov2(C,T+1)` true in Line 8. The *inertial* rules for `cov2/2` in Lines 10–11 are analogous to the ones of `cov1/2` in Lines 6–7. The final possibility is formalized by Line 12: there is an end of a RMI falsifying `cov2(C,T+1)` but making `cov1(C,T+1)` true again. Note that according to answer set semantics, any other instances of `cov1(C,T)` and `cov2(C,T)` are *false by default* and need not be specified by any rules.

Finally, Listing 4 sets the scene for solving the optimization problem in question. Firstly, in Line 2, more than two overlapping RMIs are banned in the spirit of Lemma 3. This is also why predicates `cov1/2` and `cov2/2` are sufficient to keep track of overlaps in the first place. Secondly, the objective function penalizes for under servicing (time steps not covered) and over servicing (time steps covered twice) on equal basis in Lines 5 and 7, respectively. For the sake of illustration, we have depicted two globally optimal schedules for the 8-component machine from Listing 1. The one in Fig. 4a is based on the minimization of under-coverage (red cells) only. This leads to a substantial overlap of RMIs and over-coverage indicated in blue. A periodic pattern of 10 time steps seems to be emerging, al-

Listing 3. PMS encoding: counting overlap of intervals

```
% Component-specific coverage of time steps by RMIs
cov1(C,0) :- comp(C,R,L), L>0.
cov1(C,T) :- not cov1(C,T-1), not cov2(C,T-1),
             serv(C,T), not emi(C,T), time(T-1).

cov1(C,T+1) :- cov1(C,T), not serv(C,T+1), not emi(C,T+1), time(T+1).
cov1(C,T+1) :- cov1(C,T), serv(C,T+1), emi(C,T+1), time(T+1).
cov2(C,T+1) :- cov1(C,T), serv(C,T+1), not emi(C,T+1), time(T+1).

cov2(C,T+1) :- cov2(C,T), not serv(C,T+1), not emi(C,T+1), time(T+1).
cov2(C,T+1) :- cov2(C,T), serv(C,T+1), emi(C,T+1), time(T+1).
cov1(C,T+1) :- cov2(C,T), not serv(C,T+1), emi(C,T+1), time(T+1).
```

Listing 4. PMS encoding: constraints and objective function

```
% Deny (excessive) overlaps of RMIs
:- cov2(C,T), serv(C,T+1), not emi(C,T+1), time(T+1).

% Optimization with respect to miscoverage
#minimize {1,C,T: not cov1(C,T), not cov2(C,T),
                  comp(C), time(T), T>0;
           1,C,T: cov2(C,T), comp(C), time(T)      }.
```

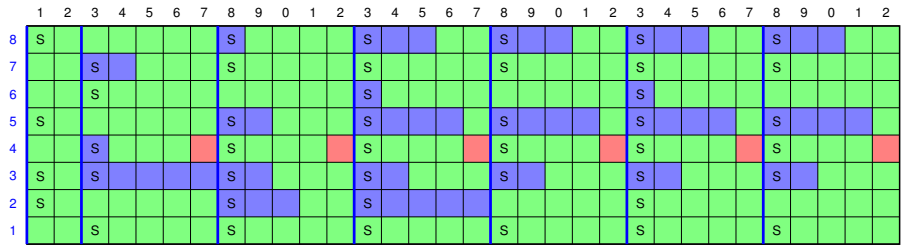
though the overall applicability of the preventive maintenance actions involved remains open in the future due to finite time horizon. However, if over-coverage is penalized equally, far better schedules result as shown in Fig. 4b. The risks from under service are slightly higher but over servicing resulting from too frequent preventive maintenance actions are decreased substantially. The number of individual preventive maintenance actions is also decreased from 44 to 36. Interestingly, the dimensions of the schedules reflect the space complexity of ground logic programs obtained from our encoding (cf. Lemma 6). The effect of the bound b on the size of the ground program is negligible.

Proposition 1. *The size of the ground program resulting from Listings 2–4 is* $\mathcal{O}(|C|h)$ *for a set of components* C *and the time horizon* h*.*

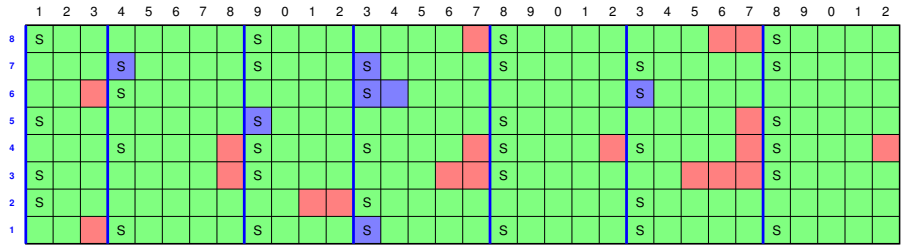
Theorem 4. *Let* $\mathcal{M} = \langle C, \iota, \rho \rangle$ *be a machine,* $\langle h, \ell, b \rangle$ *the triple of scheduling parameters, and* $P_{\mathcal{M}}$ *their representation as a ground logic program based on Listings 2–4 and a set of facts encoding* $\mathcal{M}$*.*

1. *If* X *is an (optimal) answer set of* $P_{\mathcal{M}}$*, then there is an (optimal) solution* $S_X = \langle h, \ell, b, A_X \rangle$ *to the MISCOVERAGE PMS problem* $\langle \mathcal{M}, \langle h, \ell, b \rangle \rangle$*.*
2. *If a schedule* $S = \langle h, \ell, b, A \rangle$ *is an (optimal) solution to the MISCOVERAGE PMS problem* $\langle \mathcal{M}, \langle h, \ell, b \rangle \rangle$*, then there is an (optimal) answer set* X_S *of* $P_{\mathcal{M}}$*.*

Proof (Sketch). Due to space restrictions, we concentrate on describing the one-to-one correspondence between answer sets and schedules as follows. Firstly, given an answer set X of $P_{\mathcal{M}}$, the respective PMS is $S_X = \langle h, \ell, b, A_X \rangle$ where for a component $c \in C$ and a time step $1 \leq i \leq h$, $A_X(c, i) = 1 \iff \texttt{serv}(c, i) \in X$.



(a) Schedule for $b = 7$ (Under-coverage)



(b) Schedule for $b = 7$ (Miscoverage)

Fig. 4. Globally optimal schedules for the machine of Listing 1: blue lines indicate scheduled maintenance breaks and individual preventive maintenance actions are marked with letter "`S`"

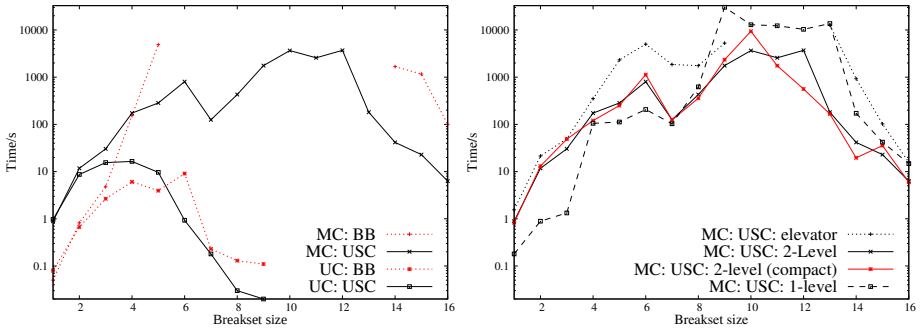
Secondly, given a PMS $S = \langle h, \ell, b, A\rangle$ for $\mathcal{M}$ subject to scheduling parameters $\langle h, \ell, b\rangle$, we may calculate for each component and a time step $0 \leq i \leq h$,

$$cnt(c, i) = |\{j \in B_c \mid j \leq i < j + \rho(c)\}| + |\{1 \mid \iota(c) > 0, i \leq \iota(c)\}|, \qquad (1)$$

i.e., the number of recommended maintenance intervals covering i while maintaining $c \in C$. Based on these, the respective answer set X_S of $P_{\mathcal{M}}$ contains

- $\texttt{comp}(c, \rho(c), \iota(c))$ and $\texttt{comp}(c)$ for every $c \in C$;
- $\texttt{time}(i)$ for every $0 \leq i \leq h$;
- $\texttt{break}(i)$ for every $1 \leq i \leq h$ such that $A(c, i) = 1$ for some $c \in C$;
- $\texttt{serv}(c, i)$ for every $c \in C$ and $1 \leq i \leq h$ such that $A(c, i) = 1$;
- $\texttt{emi}(c, i{+}\rho(c))$ for every $c \in C$ and $1 \leq i \leq h$ with $A(c, i) = 1$ and $i{+}\rho(c) \leq h$;
- $\texttt{emi}(c, \iota(c) + 1)$ for every $c \in C$ with $\iota(c) > 0$ and $\iota(c) + 1 \leq h$;
- $\texttt{cov1}(c, i)$ for every $c \in C$ and $0 \leq i \leq h$ such that $cnt(c, i) = 1$; and
- $\texttt{cov2}(c, i)$ for every $c \in C$ and $1 \leq i \leq h$ such that $cnt(c, i) = 2$.

The idea is that $X_{(S_X)} = X$ and $S_{(X_S)} = S$ hold in the bijective correspondence. Moreover, the measures $uc(c) = |\{i \mid \texttt{cov1}(c, i) \notin X, \texttt{cov2}(c, i) \notin X\}|$ and $oc(c) = |\{i \mid \texttt{cov2}(c, i) \in X\}|$ can be read off from answer sets X. Thus, the minimality of $mc(C) = uc(C) + oc(C)$ coincides with the optimality of X. □



(a) Under vs. miscoverage; USC vs. BB (b) Alternative MC encodings

Fig. 5. Runtimes in seconds for $b = 1 \ldots 16$ on logarithmic scale

It is known that function problems corresponding to optimization problems—formalized in terms of disjunction-free logic programs as above—are $\mathrm{FP}^{\mathrm{NP}}$-complete function problems [24], i.e., only polynomially many calls to an NP-oracle are needed in the worst case. As a consequence, the computational complexities of MISCOVERAGE PMS and UNDER-COVERAGE PMS are bounded from above and the corresponding decision problems reside in Δ_2^P.

Corollary 1. *Given a machine $\mathcal{M} = \langle C, \iota, \rho \rangle$ and the scheduling parameters $\langle h, \ell, b \rangle$, the respective function problems MISCOVERAGE PMS and UNDER-COVERAGE PMS for computing optimal schedules are in* $\mathrm{FP}^{\mathrm{NP}}$.

Hardness results in this respect are left for future work.

5 Experiments

In this section, we evaluate the performance of the encoding from Listings 2–4. In the experiments, we use Gringo (v. 5.2.2) as the grounder and Clasp (v. 3.3.6) as the solver. All runs are executed on an Intel(R) Xeon(R) Gold 6248 CPU with a 2.50GHz clock rate under Linux operating system.

Our preliminary screening showed that the 8-component machine from Listing 1 is already sufficient to create great variance with respect to runtimes. Thus, we concentrate on analyzing the performance of Clasp on this particular instance when the number of scheduled maintenance breaks b is increased from 1 to 16 for schedules in a scheduling timeline $h = 32$. The time required for grounding the encoding is negligible and omitted altogether.

The runtime behavior of Clasp as the back-end solver can be inspected from Fig. 5. It turns out that the minimization of under-coverage (UC) is quite easy. The lowest two plots in Fig. 5a concern optimization according to two different strategies based *branch-and-bound* (BB) and *unsatisfiable cores* (USC). For

small values of b, the BB strategy is faster but becomes slower than USC when almost fully covered schedules become feasible at $b = 8$. The upper two plots in Fig. 5a relate to the minimization of miscoverage (MC) which seems to be a far more difficult task from the computational point of view. Now the USC strategy performs much better. We think that this is due to the fact that USC approaches optimal solutions from below and since the values of the objective function are relatively small in this example, the optimal value can be reached soon. The BB strategy, however, uses upper bounds and finally, when the optimality of a found schedule is to be proved, a potentially high number of other candidates—not improving the objective value—must be excluded by the solver. Indeed, the last stage of optimization dominates in the BB strategy, and the runtimes for $b = 6 \ldots 13$ are clear outliers that reside beyond the range visible in Fig. 5a.

We have developed several variants of the ASP encoding (**'2-level'**) from Section 4 and include results for some of them in Fig. 5b. The first alternative encoding (**'2-level (compact)'**) expresses predicates `cov1(C,T)` and `cov2(C,T)` using a single predicate `cov(C,T,N)` for `N=1..2` and thus amalgamates some repeated rules. However, the performance gets slightly worse despite compaction. Yet another encoding (**'1-Level'**) infers coverage and over-coverage information *straight* from preventive maintenance actions, e.g., if `serv(C,T)` is made true (Line 8) then `cov(C,T)`, `cov(C,T+1)`, ..., `cov(C,T+R-1)` are inferred for the length `R` of the RMI. Analogous rules for over-coverage `ocov/2` are no longer linear in scheduling timeline h. It performs very well for small values of b, but degrades soon so that clearly more time is required for values $b = 9, \ldots, 14$. Finally, we mention our initial encoding (**'Elevator'**) based on up-and-down *counting* of the number of overlapping maintenance intervals in direct analogy to (1). In Listing 3, the predicates `cov1/2` and `cov2/2` are the counterparts of `cnt(C,T,1)` and `cnt(C,T,2)` used in that encoding, and there is no pendant for `cnt(C,T,0)` expressible via default negation, i.e., if `not cov1(C,T)` and `not cov2(C,T)` hold simultaneously. The respective plot in Fig. 5b indicates that such a systematic saving in the number of predicates pays off.

To further assess the scalability of the method, we tested the 2-Level encoding of miscoverage and the USC strategy with 8 scheduled maintenance breaks and 10 randomly generated machines per each of the sizes from 1 to 16 components. Fig. 6 shows the averages of the running times in this experiment. When optimizing the miscoverage, a majority of the larger machine sizes hit the timeout of 32800 s. This highlights that without approximations and further optimizations, the problem cannot be solved on a large scale.

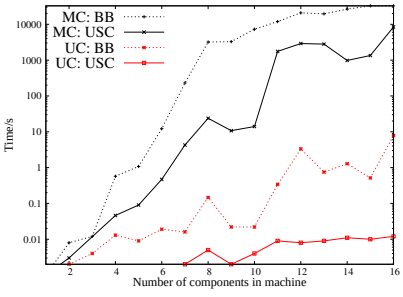


Fig. 6: Effect of machine size

6 Conclusion

This paper studies PMS of continuously operating, rotating multi-component machines and formalizes the PMS problem as a search problem that abstracts away from side-constraints and stochastic aspects of PMS. The focus is on globally optimal maintenance scheduling under discrete-valued optimization criteria.

Although the efficiency of global optimization techniques has been constantly improving, the problem formalization, encoding and optimization techniques can make a major difference in the feasibility of the technological approach. We present an ASP encoding for the EXACT MISCOVERAGE PMS problem and carry out a preliminary feasibility study for it, according to which the quest for performance improvement remains substantial if we extend the breakset size or the machine size. Our problem sizes are on a par with previous research [12, 26], but a precise comparison is omitted because prior implementations of PMS are not generally available and they differ substantially in their concepts and parameters. Nevertheless, we show that the EXACT MISCOVERAGE PMS problem is one of the NP-hard (scheduling) problems [13, 25]. In addition, by putting our encodings publicly available[2], we facilitate their improvement in the future. New kinds of encodings could be devised, e.g., by using recent extensions based on temporal operators [5] or multi-shot solving [15].

It is worth noting that although we compute globally optimal schedules, the same formalization can be approached with local or approximate optimization techniques. It is also within our interests to study extensions of the PMS problem. Hard side-constraints such as availability and resource constraints are easy to incorporate into ASP encodings in an orthogonal way, and the EXACT MISCOVERAGE PMS problem can be embedded into a modularized framework for condition-based rescheduling. See, e.g., [27, 28] where the recommended maintenance intervals depend on the workload or the current condition of the machine.

Acknowledgments. The support from the Academy of Finland within the project AI-ROT (#335718) is gratefully acknowledged.

References

1. Ali, M.B., Sassi, M., Gossa, M., Harrath, Y.: Simultaneous scheduling of production and maintenance tasks in the job shop. International Journal of Production Research **49**, 3891–3918 (2011). https://doi.org/10.1080/00207543.2010.492405
2. Alviano, M., Dodaro, C., Leone, N., Ricca, F.: Advances in WASP. In: LPNMR 2015. pp. 40–54 (2015). https://doi.org/10.1007/978-3-319-23264-5_5
3. Banbara, M., Inoue, K., Kaufmann, B., Okimoto, T., Schaub, T., Soh, T., Tamura, N., Wanko, P.: *teaspoon*: solving the curriculum-based course timetabling problems with answer set programming. Ann Oper Res **275**, 3–37 (2019). https://doi.org/10.1007/s10479-018-2757-7
4. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Commun. ACM **54**(12), 92–103 (2011). https://doi.org/10.1145/2043174.2043195

[2] https://github.com/asptools/benchmarks

5. Cabalar, P., Kaminski, R., Morkisch, P., Schaub, T.: telingo = ASP + time. In: LPNMR 2019. pp. 256–269 (2019). https://doi.org/10.1007/978-3-030-20528-7_19
6. Cassady, C., Murdock, P., Pohl, E.: Selective maintenance for support equipment involving multiple maintenance actions. EJOR **129**(2), 252–258 (2001), a Global View of Industrial Logistics. https://doi.org/10.1016/S0377-2217(00)00222-8
7. Chansombat, S., Pongcharoen, P., Hicks, C.: A mixed-integer linear programming model for integrated production and preventive maintenance scheduling in the capital goods industry. Int J Prod Res **57**(1), 61–82 (2019). https://doi.org/10.1080/00207543.2018.1459923
8. Chen, X., An, Y., Zhang, Z., Li, Y.: An approximate nondominated sorting genetic algorithm to integrate optimization of production scheduling and accurate maintenance based on reliability intervals. J Manuf Syst **54**, 227–241 (2020). https://doi.org/10.1016/j.jmsy.2019.12.004
9. Do, P., Vu, H.C., Barros, A., Bérenguer, C.: Maintenance grouping for multi-component systems with availability constraints and limited maintenance teams. Reliability Engineering & System Safety **142**, 56–67 (2015). https://doi.org/10.1016/j.ress.2015.04.022
10. Dodaro, C., Maratea, M.: Nurse scheduling via answer set programming. In: LPNMR 2017. pp. 301–307. Springer (2017). https://doi.org/10.1007/978-3-319-61660-5_27
11. Eiter, T., Geibinger, T., Musliu, N., Oetsch, J., Skocovský, P., Stepanova, D.: Answer-set programming for lexicographical makespan optimisation in parallel machine scheduling. In: KR 2021. pp. 280–290 (2021). http://dx.doi.org/10.24963/kr.2021/27
12. Frost, D., Dechter, R.: Optimizing with constraints: A case study in scheduling maintenance of electric power units. In: CP 1998. pp. 469–469. Springer (1998). https://doi.org/10.1007/3-540-49481-2_40
13. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Company (1979)
14. Gebser, M., Kaminski, R., Kaufmann, B., Romero, J., Schaub, T.: Progress in clasp series 3. In: LPNMR 2015. pp. 368–383 (2015). https://doi.org/10.1007/978-3-319-23264-5_31
15. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Multi-shot ASP solving with clingo. Theory Pract. Log. Program. **19**(1), 27–82 (2019). https://doi.org/10.1017/S1471068418000054
16. Gebser, M., Kaminski, R., Ostrowski, M., Schaub, T., Thiele, S.: On the input language of asp grounder gringo. In: Erdem, E., Lin, F., Schaub, T. (eds.) Logic Programming and Nonmonotonic Reasoning. pp. 502–508. Springer (2009). https://doi.org/10.1007/978-3-642-04238-6_49
17. Geurtsen, M., Didden, J.B., Adan, J., Atan, Z., Adan, I.: Production, maintenance and resource scheduling: A review. EJOR (2022). https://doi.org/10.1016/j.ejor.2022.03.045
18. Hoai, M.T., Luong, H.T.: Selective maintenance policy with time-window constraint. In: Proceedings of the 7th Asia Pacific Industrial Engineering and Management Systems Conference 2006. Bangkok, Thailand (2006)
19. Nguyen, K.A., Do, P., Grall, A.: Condition-based maintenance for multi-component systems using importance measure and predictive information. International Journal of Systems Science: Operations & Logistics **1**(4), 228–245 (2014). https://doi.org/10.1080/23302674.2014.983582

20. Nguyen, K.A., Do, P., Grall, A.: Multi-level predictive maintenance for multi-component systems. Reliability Engineering & System Safety **144**, 83–94 (2015). https://doi.org/10.1016/j.ress.2015.07.017
21. Olde Keizer, M., Flapper, S., Teunter, R.: Condition-based maintenance policies for systems with multiple dependent components: A review. EJOR **261**(2), 405–420 (2017). https://doi.org/10.1016/j.ejor.2017.02.044
22. Rajaprasad, S.V.S.: Investigation of reliability, maintainability and availability of a paper machine in an integrated pulp and paper mill. International Journal of Engineering, Science and Technology **10**(3), 43–56 (2018). https://doi.org/10.4314/ijest.v10i3.5
23. Sachdeva, A., Kumar, D., Kumar, P.: Planning and optimizing the maintenance of paper production systems in a paper plant. Computers & Industrial Engineering **55**, 817–829 (2008). https://doi.org/10.1016/j.cie.2008.03.004
24. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence **138**(1-2), 181–234 (2002). https://doi.org/10.1016/S0004-3702(02)00187-X
25. Ullman, J.: NP-complete scheduling problems. JCSS **10**(3), 384–393 (1975). https://doi.org/10.1016/S0022-0000(75)80008-0
26. You, M.Y., Meng, G.: A modularized framework for predictive maintenance scheduling. Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability **226**(4), 380–391 (2012). https://doi.org/10.1177/1748006X11431209
27. Youssef, H., Brigitte, C.M., Noureddine, Z.: Lower bounds and multiobjective evolutionary optimisation for combined maintenance and production scheduling in job shop. In: IEEE 2003 Conference on EFTA. vol. 2, pp. 95–100 (2003). https://doi.org/10.1109/ETFA.2003.1248675
28. Zheng, Y., Lian, L., Mesghouni, K.: Comparative study of heuristics algorithms in solving flexible job shop scheduling problem with condition based maintenance. Journal of Industrial Engineering and Management **7**(2), 518–531 (2014). http://dx.doi.org/10.3926/jiem.1038
29. Zurn, H., Quintana, V.: Generator maintenance scheduling via successive approximations dynamic programming. IEEE Trans on Power Apparatus and Systems **94**(2), 665–671 (1975). https://doi.org/10.1109/T-PAS.1975.31894
30. Öhman, M., Hiltunen, M., Virtanen, K., Holmström, J.: Frontlog scheduling in aircraft line maintenance: From explorative solution design to theoretical insight into buffer management. Journal of Operations Management **67**(2), 120–151 (2021). https://doi.org/10.1002/joom.1108